REMARKS

Applicants have amended the title to conform to the title of the application as filed and as indicated in the original filing receipt.

Claims 1-21 are pending and stand rejected by the examiner. Claims 1, 12 and 21 are independent claims. Applicants have canceled claims 13 and 14 and amended claims 12, 15 and 17. No new matter was added.

The examiner uses Bates and Wen to reject claims 1-7, 11 and 21 as having been obvious. Applicants disagree and believe that the examiner has mischaracterized the references.

For example, claims 1 and 21 recite "receiving a program instruction and an identification representing a selected one of the plurality of microengines from a remote user interface connected to the processor, pausing program execution in the threads executing in the selected microengine," or similar language. The cited references at least fail to teach or suggest this quoted feature. Bates merely allows a programmer to statically insert one or more thread identification control points and breakpoints into a non-executing program, much like a standard debugger does. When a Bates' thread identification control point is then encountered by an executing program, a thread identification is stored. When a breakpoint is subsequently encountered, program execution ceases and a user can then request to view the thread or threads associated with the one or more encountered thread identification control points before program cessation. More specifically, Bates discloses:

> after program execution is halted by a break point, the user may request retrieval of the thread identifiers that encountered a certain statement number. Moreover, in the illustrative embodiment, the program element is associated with threads by implementing a thread identification control point in a manner similar to conventional break points at the address for the program element.
>
> Thread identification control points and break points are both examples of control points, which are generally user inserted interruptions to program execution. In the illustrative embodiment, an invalid instruction is inserted at an address corresponding to a statement used in a program. Thus, the invalid instruction causes a hardware interrupt, also referred to as a system exception. Other control point implementations may be used in the alternative. (Col. 3, lines 24-31)

Bates merely teaches control points that are inserted prior to program execution like a standard debugger. A first Bates control point is inserted to identify a thread when a program is executing and a second control point stops program execution and allows the user to view the thread associated with first control point. This is very different from receiving a program instruction and an identification representing a selected one of the plurality of microengines from a remote user interface connected to the processor and pausing program execution in the threads executing in the selected microengine.

Wen fails to make up for the deficiency in Bates. Wen merely teaches monitoring a single processor (the "master processor") and distributing software for indirectly monitoring the remaining processor or processors (the "slave processors") in a memory which is shared by the master processor and the slave processors. More specifically, Wen discloses:

> During application software development, host monitor software 34 (which is resident in the host computer 10), together with control monitor software 36 (which is resident in the control processor 12), monitor each node 14. They monitor the master processor 22 of the node 14 directly through the master transmit/receive buffer 20 for the node 14. The node's master transmit/receive buffer 20 is connected between the VME bus 16 and the master processor 22.
>
> The slave processors 24 are not connected directly to the VME bus 16 by any transmit/receive buffer. Instead, each slave processor 24 and the master processor 22 communicate with each other through a slave transmit/receive buffer 38 in the RAM portion 28 of the shared memory 26. These slave transmit/receive buffers 38 are operated by master monitor software 40 and slave monitor software 42. The master monitor software 40 and slave monitor software 42 are also resident in the
> shared memory 26. They thereby provide the host computer 10 with distributed (indirect) monitoring of the node's slave processors 24 through the control processor 12 and the node's master processor 22. Additional transmit/receive buffers may be provided between (or among) slave processors 24 if convenient, but there must be a slave transmit/receive buffer 38 between each slave processor 24 and the master processor 22. (Col. 3, lines 4-28)

Accordingly, claims 1 and 21 are rendered obvious by Bates and Wen.

The examiner uses Bates, Wen and Mitchell to reject claims 8-10 and 12-20 as having been obvious.

Applicants disagree. Claim 12 recites "perform a breakpoint routine residing in a debug library in one of the contexts, a breakpoint in the context pointing to the breakpoint routine, the breakpoint inserted into the context in response to a user request received through a remote user interface connected to the processor, the request including a context identification." The cited references at least fail to teach or suggest this quoted feature.

As described above with reference to claims 1 and 21, Bates and Wen fail to teach or suggest the breakpoint inserted into the context in response to a user request received through a remote user interface connected to the processor, the request including a context identification. Mitchell fails to provide for the deficiencies in Bates and Wen. Mitchell merely teaches a data processor in which an embedded emulator communicates with a control emulation system using a serial communications link involving one pin of the data processor package. More specifically, Mitchell discloses:

> The data processor having an emulator according to the present invention performs emulation under software control. In order to achieve this a reserved memory containing emulation instructions is provided as an integral part of the data processor. (Col. 4, lines 56-60)

This is very different from performing a breakpoint routine residing in a debug library in one of the contexts, a breakpoint in the context pointing to the breakpoint routine, the breakpoint inserted into the context in response to a user request received through a remote user interface connected to the processor, the request including a context identification. Accordingly, claim 12 is not rendered obvious by Bates, Wen and Mitchell.

It is believed that all of the pending claims have been addressed. However, the absence of a reply to a specific rejection, issue or comment does not signify agreement with or concession of that rejection, issue or comment. In addition, because the arguments made above may not be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper, and the amendment of any claim does not necessarily signify concession of unpatentability of the claim prior to its amendment.

Please apply any charges or credits to deposit account 06-1050.

Respectfully submitted,

Date: May 10, 2004

Kenneth F. Kozik
Reg. No. 36,572

Fish & Richardson P.C.
225 Franklin Street
Boston, MA 02110-2804
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

20829711.doc